Estructura de Datos y Algoritmos

Grados de la Facultad de Informática de la UCM (Grupos C y F). Curso 2016-2017
Examen final de septiembre **Tiempo: 3 horas**

Ejercicio 1 [2.5 puntos]

Sea un vector $V[0..N), N \geq 1$ de enteros tales que dos elementos cualesquiera consecutivos no son iguales. Se dice que forma un *valle* si existe una posición tal que todos los elementos anteriores a ella forman una secuencia estrictamente decreciente y a partir de la misma todos los elementos son estrictamente creciente:

Formalmente:

$$\{N \geq 1 \wedge strict(V, 0, N)\}$$
$$\textbf{fun } isValley(\ \mathbf{V[0..N)}\ of\ \textbf{int},\ int\ \mathbf{N})\textbf{dev } b:\ \textbf{bool}$$
$$\{b = \exists n : 0 \leq n < N : valley(0, n, N)\}$$

donde

$$strict(V, 0, N) \equiv (\forall i : 0 \leq i < N - 1 : V[i] \neq V[i + 1])$$
$$valley(0, n, N) \equiv strictDec(V, 0, n) \wedge strictInc(V, n, N)$$
$$strictDec(V, 0, n) \equiv \forall i : 0 \leq i < n - 1 : V[i] > V[i + 1])$$
$$strictInc(V, n, N) \equiv \forall i : n \leq i < N - 1 : V[i] < V[i + 1]$$

- **0.25** Define el predicado $strict(V, 0, N)$.
- **2p** Programar un procedimiento iterativo que dado un vector de enteros de entrada determine si los valores de éste determinan una configuración de *valle*.
- **0.25** Justificar su complejidad.

Entrada	Salida
1 1	1
3 1 2 3	1
3 3 2 1	1
4 3 2 1 2	1
5 3 2 1 2 1	0

Ejercicio 2 [2.5 puntos]

El juego de cartas *WhiteJacket* es cada vez más popular. Su funcionamiento es como sigue: Juega solo un jugador y hay dos barajas de cartas en la mesa. En cada jugada hay tres opciones, coger una carta del montón izquierdo, coger una carta del montón derecho, o plantarse, en cuyo caso acaba el juego. Se empieza con 0 puntos. Coger una nueva carta (de cualquiera de los montones) resta n puntos, siendo n el número de jugada (la primera jugada resta un punto, la segunda resta dos puntos, etc.), y también suma p puntos siendo p el valor numérico de la carta (siempre positivo).

Se pide implementar un algoritmo que encuentre la secuencia de jugadas con mejor puntuación y su puntuación asociada. En caso de empate se debe elegir la secuencia más corta, y en caso de empate en longitud la que se encuentre primero, entendiendo que el algoritmo explorará antes las jugadas que cogen del montón izquierdo. La secuencia debe devolverse como un `List<bool>`, siendo `false` coger del montón izquierdo y `true` coger del derecho.

La función principal proporcionada para hacer pruebas lee los dos montones en sendos arrays (secuencias de enteros no negativos acabando cada lectura con un -1 que no se incluye en el array), llama a la función pedida, y muestra por pantalla la mejor puntuación y en la siguiente línea la mejor secuencia (ver ejemplos). El proceso se repite hasta introducir un array vacío (es decir, un -1).

Entrada	Salida
2 4 -1 3 1 -1	3 0 0
1 5 -1 7 4 -1	8 1 1
1 -1 1 -1	0
1 2 -1 2 2 -1	1 1

Ejercicio 3 [2.5 puntos]

Extiende el TAD `Cola` visto en clase (`Queue.h`) con una nueva operación interna y pública cuya cabecera en `C++` es

```
void cuela(const T& a, const T& b);
```

que mueve al elemento `b` de su posición a la posición inmediatamente detrás del elemento `a`. En caso de haber múltiples apariciones de los elementos `a` y/o `b` se considerará: la primera aparición de `a`, y la primera aparición de `b` tras la primera aparición de `a`. Si alguno de los elementos no se encuentra en la cola, o bien, si `b` no aparece detrás de `a`, la operación no tendrá efecto. Indica y justifica la complejidad de la operación implementada. *Requisitos:* No se puede crear ni destruir memoria dinámica, ni tampoco modificar los valores almacenados en la cola.

Entrada	Salida
1 2 3 4 -1 1 3	1 3 2 4
1 2 3 4 -1 1 4	1 4 2 3
1 2 3 4 -1 2 1	1 2 3 4
3 1 2 1 3 4 -1 1 3	3 1 3 2 1 4

La función principal proporcionada para hacer pruebas lee la cola de enteros (secuencia de enteros no negativos acabando la lectura con un -1 que no se incluye en la cola), después los enteros `a` y `b`, llama a la función pedida, y muestra por pantalla la cola resultante (ver ejemplos). El proceso se repite hasta introducir una cola vacía (es decir, un -1).

Ejercicio 4 [2.5 puntos]

Apartado a) Extiende el TAD `TreeMap` visto en clase con una nueva operación interna y pública de nombre `keyInBounds`, que recibe una clave `k` y devuelve un booleano que será `true` si y solo si en la tabla existe alguna clave menor o igual que `k` y también alguna mayor o igual. Justifica la complejidad de la operación implementada.

Apartado b) Implementa una operación con igual nombre y comportamiento a la del apartado a) pero como función externa al TAD `TreeMap`, que por tanto recibe como parámetros un `TreeMap<K,V>` y una clave, y devuelve un booleano. Justifica la complejidad de la operación implementada.

Entrada	Salida
5 2 8 -1 3	si
5 2 8 -1 9	no
8 2 5 -1 7	si
8 2 5 -1 9	no

La función principal proporcionada para hacer pruebas en ambos apartados hace lo siguiente: construye un `TreeMap<int,Nada>` a partir de una secuencia de enteros no negativos que se van insertando, acabando la lectura con un -1 que no se inserta, después lee la clave `k` (un entero), llama al método o a la función pedida, y muestra por pantalla el resultado obtenido en el formato "si" o "no" (ver ejemplos). El proceso se repite hasta introducir una secuencia vacía (es decir, un -1).

Instrucciones

- Entrega un fichero .cpp para cada ejercicio, nombrado ejerX.cpp siendo X el número del ejercicio, excepto para el ejerc. 3 y el apartado a) del 4 en los que se entregará el Queue.h y TreeMap.h resp.
- Al principio de cada fichero debe aparecer, en un comentario, vuestro nombre y apellidos, dni y puesto de laboratorio. También debéis incluir unas líneas explicando qué habéis conseguido hacer y qué no.
- Todo lo que no sea código C++ (explicaciones, especificaciones, invariantes, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
- Los TADs vistos y las plantillas para poder probar vuestras soluciones se obtienen pulsando en el icono del Escritorio "Publicacion docente ...", después en "Alumno recogida docente", y en el programa que se abre, abriendo en la parte derecha la carpeta EDA-F, arrastrando los ficheros a hlocal (en la izqda).
- La entrega se realiza pulsando en el icono del escritorio "Examenes en Labs ...", y posteriormente, utilizando el programa que se abre, colocando los ficheros a entregar en la carpeta de vuestro puesto (en el lado derecho).